A hybrid approach to the verification of computer interpretable guidelines

A Hybrid Approach to the Verification of Computer Interpretable Guidelines

Luca Anselma[3], Alessio Bottrighi[1], Laura Giordano[1], Arjen Hommersom[4], Gianpaolo Molino[2], Stefania Montani[1], Paolo Terenziani[1], Mauro Torchio[2]

[1] DiSIT, Università del Piemonte Orientale "Amedeo Avogadro", Viale Teresa Michel 11, 15121 Alessandria, Italy
[2] Azienda Ospedaliera San Giovanni Battista, corso Bramante 88, 10126 Torino, Italy
[3] Dipartimento di Informatica, Università di Torino, corso Svizzera 185, 10149 Torino, Italy

Abstract. Computer Interpretable Guidelines (CIGs) are assuming a major role in the medical area, in order to enhance the quality of medical assistance by providing physicians with evidence-based recommendations. However, the complexity of CIGs (which may contain hundreds of related clinical activities) demands for a verification process, aimed at assuring that a CIG satisfies several different types of properties (e.g., verification of the CIG correctness with respect to several criteria). Verification is a demanding task, which may be enhanced through the adoption of advanced Artificial Intelligence techniques. In this paper, we propose a general and hybrid approach to address such a task, suggesting that, given the heterogeneous character of the knowledge in CIGs, different forms of verification should be supported, through the adoption of proper (and different) methodologies.

1 Introduction

Clinical Practice Guidelines (CPGs) can be defined as a means for specifying the "best" clinical procedures and for standardizing them. The adoption of CPGs, by supporting physicians in their decision making and diagnosing activities, may provide crucial advantages, both in individualized health care, and in the overall service offered by a health care organization. In particular, it has been shown [1] that CPGs can improve the quality of patient care, reduce variations in quality of care, and reduce costs. These observations justify the increasing number of CPGs which have been defined in the last decade, covering a large spectrum of diseases and medical procedures. Given the relevance of this phenomenon, in the last two decades a lot of efforts have been devoted in order to provided formal representations of guidelines, that can be treated by computer systems (usually called Computer Interpretable Guidelines CIGs for short). As discussed in **G.1** many approaches have focused on the development of guideline representation formalisms, and/or systems to acquire, store and execute CIGs. However, the effort in defining and disseminating CIGs has not always been coupled by a parallel effort in guaranteeing their "quality" [2]: despite the fact that CPGs and/or

CIGs are issued by recognized experts' committees, they might be ambiguous or incomplete [3], or even inconsistent. The need for guideline quality verification is thus clearly emerging. As we will show in this paper, computer-based approaches can provide crucial advantages in this context.

In particular, in this paper we suggest that, given the heterogeneous character of the knowledge contained in CIGs, different forms of verifications should be supported, demanding for an hybrid approach in which different representation formalisms are used (to properly capture different types of knowledge) and different methodologies are devised (to properly reason with the different formalisms). In particular, in this paper, we focus on three different forms of verification:

1. verification that the temporal constraints in a CIG are consistent, through constraint-based temporal reasoning techniques;
2. verification of different medical properties of a CIG (e.g., its capability of coping with a given type of patients, or to support specific types of treatments), through model checking;
3. verification of probabilistic properties of a CIG in the context of a probabilistic knowledge base, through probabilistic modelling.

2 Representing and Reasoning with Temporal Constraints

Representing and reasoning with temporal constraints is an essential feature for computer-based approaches to clinical guidelines. In particular, a temporal manager coping with time-related issues can be exploited in different ways in the management of clinical guidelines. For instance, during the acquisition of a new guideline, the consistency of the temporal constraints it contains can be automatically checked; during the execution of a guideline on a specific patient, the temporal manager can be used to check whether the specific actions have been executed in such a way that the constraints in the guideline have been respected, or to determine the times when the next actions need to be executed. However, although many domain-independent temporal managers have been devised within the Artificial Intelligence (AI) literature, and several approaches to time-related issues have been faced within the clinical guideline literature, several new challenges have to be addressed when dealing with temporal representation and temporal reasoning about clinical guidelines.

2.1 Desiderata for a CIG temporal reasoner

As in most AI approaches to the treatment of time, also in the context of CIGs we must take into account the fundamental trade-off between the expressiveness of temporal formalisms and the computational complexity of the correct and complete temporal reasoning algorithms operating on them.

While expressiveness is an obvious desideratum, we will now briefly motivate the second term of the above trade-off: correctness, completeness, and tractability. First, it is important to stress that a formalism for temporal constraints is not very useful if it is not paired with algorithms for temporal reasoning, performing temporal inferences on a set of constraints (expressed in the given formalism) and/or checking their consistency. Consider, for instance, a Knowledge Base KB containing the temporal constraints (i) and (ii) among three events A, B and C.

$$KB = \{\textit{(i) A before B}; \textit{(ii) B before C}\}$$

The constraint *(iii) A before C* can be inferred because it is logically implied by *(i)* and *(ii)*, so that, given KB, one can correctly assert *(iii)*, but not *(iv) A after C*, which is actually inconsistent with KB. In other words, the set of constraints $KB' = \{(i), (ii), (iv)\}$ cannot be satisfied. Temporal reasoning is necessary in order to support such an intended semantics. With no temporal reasoning, a CIG may contain the above set of temporal constraints, and thus be not executable (since there is no way of satisfying the constraints).

Of course, temporal reasoning algorithms are computationally expensive. An important desideratum is tractability, i.e., the fact that the running time of the algorithms grows as a fixed power of the number of the actions and/or constraints in the knowledge base (i.e., in polynomial time).

However, temporal reasoning algorithms should also be correct, i.e., such that they only infer constraints that are logically implied by the initial set of constraints (in fact, correctness grants that no wrong inference is made). Completeness (i.e., the fact that all logically implied constraints are actually inferred) is a fundamental desideratum as well, since it is essential in order to grant that the system's answers are fully reliable (e.g., if (iii) is not inferred from $\{(i), (ii)\}$, the answer to the question "Is (iv) consistent with $\{(i), (ii)\}$?" may be yes).

In particular, as in most AI approaches, the main task of our temporal reasoning algorithms is that of checking the consistency of temporal constraints in a guideline. In fact, real-world guidelines usually consist of hundreds of actions, often related by temporal constraints. This means that: (i) the fact that hundreds of constraints are mutually consistent cannot be taken for granted and (ii) consistency checking cannot be directly performed by physicians (and/or by a knowledge engineer), since making explicit all the possible implications of such a large number of constraints is an overwhelming and too complex task.

Dealing with temporal constraints in clinical guidelines: new challenges and open problems Despite the large amount of valuable works, there still seems to be a gap between the range of phenomena covered by current AI constraint-based approaches and the needs arising from clinical guidelines management. In essence, while many AI approaches to temporal constraints are focused on the treatment of a specific type of constraints only (e.g., qualitative temporal constraints), in the CIG context several different issues and types of constraints need to be taken into account:

(i) qualitative (e.g., at the same time) and quantitative (e.g., at least ten days after) constraints between actions;
(ii) repeated/periodic events (and constraints between them);
(iii) all the above types of constraints may be imprecise and/or partially defined;
(iv) temporal constraints involved by part-of relations between actions in the CIGs
(v) the distinction between (temporal constraints between) classes of actions (e.g., an action in a general guideline) and instances of such actions.

As regards issue (iv), notice that most CIG formalisms support multiple levels of abstraction, through the definition of composite actions, and the specification of their components. However, part-of decomposition involves temporal constraints, since each composite action temporally contains its components. Finally, issue (v) points out that actions in CIGs can be conceived as classes of actions, which admit multiple instantiations, whereas CIGs are applied to specific patients. This involves the treatment of some form of temporal constraint inheritance from classes to instances. As a real example of the temporal complexity of the CIG domain, consider Example 1 (which is a simplified part of a guideline about multiple myeloma).

Example 1. The therapy for multiple myeloma is made by six cycles of 5-day treatment, each one followed by a delay of 23 days (for a total time of 24 weeks). Within each cycle of 5 days, 2 inner cycles can be distinguished: the melphalan treatment, to be provided twice a day, for each of the 5 days, and the prednisone treatment, to be provided once a day, for each of the 5 days. These two treatments must be performed in parallel.

Temporal constraints such as the ones in Example 1 are challenging for the constraint-based formalisms developed within the AI literature.

Obviously, the interplay between issues (i)-(v) needs to be dealt with, too. For example, the interaction between composite and periodic events might be complex to represent and manage. In fact, in the case of a composite periodic event, the temporal pattern regards the components, which may, recursively, be composite and/or periodic events. For instance, consider Example 1. In Example 1, the instances of the melphalan treatment must respect the temporal pattern "twice a day, for 5 days", but such a pattern must be repeated for six cycles, each one followed by a delay of 23 days, since the melphalan treatment is part of the general therapy for multiple myeloma.

While some of the above issues have been treated in an ad-hoc way in the literature, in our approach we aim at devising a general module coping in an integrated way with all of them. The temporal knowledge server will act as an independent module and the temporal problems in different clinical guidelines will be delegated to such a server. The strategy we chose to adopt in order to achieve our goal is that of devising a two-layer approach:

1. the high-level layer provides a high-level language to represent the above-mentioned temporal phenomena and to offer several temporal reasoning facilities;

2. the low-level layer consists of an internal representation of the temporal constraints, on which temporal constraint propagation algorithms operate.

We designed our high-level language with specific attention to modelling repeated actions, and in such a way that tractable temporal reasoning can be supported. At the low-level layer, we chose to exploit as much as possible STP (Simple Temporal Problem), a standard AI temporal reasoning framework [4]. In a certain sense, our approach uses STP as an "assembly language" and builds an expressive "high-level temporal reasoning framework" on top of it. Obviously, the gap between our high-level language and STP is very large. Filling such a gap is the main contribution of our approach, and has involved the design of suitable temporal reasoning algorithms to cope with issues (i)-(v) above, as well as an extension of the STP framework itself (to consider labelled trees of STPs).

2.2 High-level formalism for CIG temporal constraints

Our high-level language allows one to express temporal constraints of the different types discussed above.

Dates can be expressed by the predicate **date(A, L1, U1, L2, U2)**, stating that the action A must start between dates L1 and U1 and end between dates L2 and U2. Precise dates can be expressed imposing L1=U1 or L2=U2. Please note that also unknown dates are allowed by imposing that the extremes assume value $-\infty$ or $+\infty$. Other constructs include the predicate **duration(A, L, U)**, stating that the duration of action A must be included between L and U, **delay(P1, P2, L, U)**, stating that the delay between P1 and P2 must be between L and U, where P1 and P2 are time points (i.e., starting or ending points of actions). Also qualitative temporal constraints such as **"before", "after", "during"** are supported by our language: in fact all and only the qualitative constraints that can be mapped to conjunctions of STP constraints are supported.

For representing composite actions we support the predicate **partOf(A', A)**, stating that the action A' is part of the composite action A. Please note that the partOf relation induces a temporal constraint between the actions: i.e., action A' must be during action A. The predicates described above can be also used for representing temporal constraints between instances of actions.

In order to describe the relation between instances and classes, we need to introduce a further predicate, **instanceOf(I, A, p)** to represent the fact that the instance of action I is an instance of the class of actions A. If A is a repeated action, then p represents the fact that I is an instance of the p^{th} repetition of A (if A is not a repeated action, $p = 0$).

Regarding repetition of actions, we provide the predicate **repetition(A, RSpec)**, to state that the (possibly composite) class of action A is repeated according to the specification RSpec. *RSpec* is a recursive structure of arbitrary depth of the form

$$RSpec = \langle R_1, \ldots, R_n \rangle,$$

where each level R_i states that the actions described in the next level (i.e., R_{i+1}, or by convention the action A, if $i = n$) must be repeated a certain number of

times in a certain time span. To be more specific, any basic element R_i consists of a quadruple

$$R_i = \langle nRepetitions_i, I\text{-}Time_i, repConstraints_i, conditions_i \rangle,$$

where the first term represents the number of times that R_{i+1} must be repeated, the second one represents the time span in which the repetitions must be included, the third one may impose a pattern that the repetitions must follow, and the last one allows to express conditions that must hold so that the repetition can take place. Informally, we can roughly describe the semantics of a quadruple R_i as the natural language sentence repeat R_{i+1} $nRepetitions_i$ times in exactly $I\text{-}Time_i$, if $conditions_i$ hold.

A detailed treatment of such a specification is outside the goals of the current paper. Indeed, in [5] the expressiveness of the language for repetitions has been studied, on the basis of both the classification criteria provided by Egidi and Terenziani [6, 7] and by Bettini [8].

Additionally, in [5] the semantics of such specifications has been formally studied.

For example, the melphalan treatment in Example 1 can be represented as $Repetition(melphalan, \langle R_0 = \langle 5, 5d, , \rangle, R_1 = \langle 2, 1d, , \rangle \rangle)$, meaning that the treatment is composed by two levels: R_0 states that R_1 must be repeated five times in five days and R_1 states that melphalan must be administered twice a day.

2.3 Reasoning with temporal constraints in CIGs

Regarding the instances of actions, we designed the high-level language in such a way that all constraints can be mapped onto bounds on differences and, thus, internally represented as a "standard" STP framework [4].

However, regarding the classes of events, while dates, delays, durations and qualitative temporal constraints might be represented with an STP about classes, it is not possible to represent in such a basic way also the temporal constraints about repeated/periodic and/or composite actions. We thus introduce STP-trees, as a suitable low-level representation of temporal constraints, on which temporal reasoning algorithms can operate.

STP-tree In our approach, the overall set of constraints between actions in the CIG is represented by a tree of STPs (STP-tree henceforth). The root of the tree is the STP which represents the constraints between all the actions in the guideline, except the components of repeated actions.

The STP-tree corresponding to a guideline can be automatically constructed on the basis of the temporal constraints in the guideline (expressed using the high-level language in Subsection 2.2) by executing an algorithm which operates recursively, from the root to the leaves, by putting in each STP-node all the actions except the components of repeated actions, which are represented in separate STP-nodes. On the other hand, the partOf relations not involving repeated actions are represented in the same STP as the composite action by adding to such an STP-node the constraints that all the components are contained into the corresponding composite action.

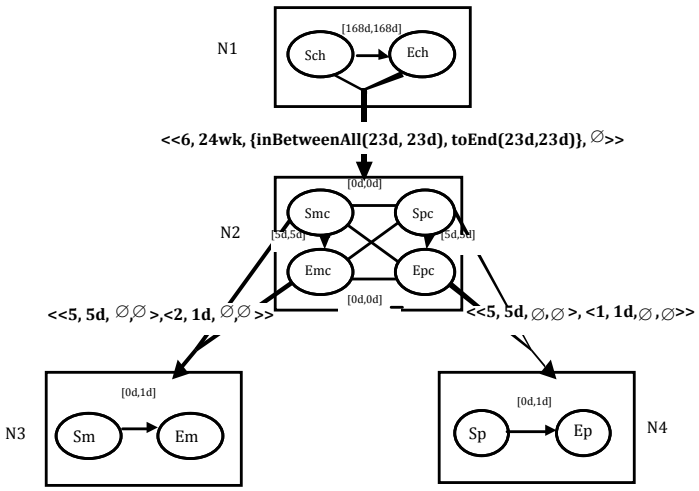


Fig. 1. STP-tree for the multiple mieloma chemotherapy guideline in Example 1. Thin lines and arcs between nodes in a STP represent bound on differences constraints. Arcs from a pair of nodes to a child STP represent repetitions. Arcs between any two nodes X and Y in a STP of the STP-tree are labeled by a pair [n,m] representing the minimum and maximum distance between X and Y. Sch, Ech, Smc, Emc, Spc, Epc, Sm, Em, Sp and Ep stand for the starting (S) and ending (E) points of chemotherapy, melphalan cycle, prednisone cycle, melphalan treatment and prednisone treatment, respectively.

To summarize, in the STP-tree there are as many STP-nodes as the number of repeated actions, and in each STP-node there are as many actions as the number of actions in the guideline that are parts of the repeated action that the STP-node represents. Specifically, each action is represented in the STP-node as a pair of time points, while constraints between (not repeated) actions are represented by arcs connecting them.

For instance, in Figure 1, we show the STP-tree representing (at the low-level) the temporal constraints in Example 1.

Additionally, an independent STP must be used in order to represent the temporal constraints about the specific instances of the actions of the guidelines, as emerging from executions of the guidelines on specific patients.

Checking the consistency of a guideline Given an STP-tree, it is possible to check its consistency in an intensional way, i.e., without generating every repetition of repeated actions. However, it is not sufficient to check the consistency of each STP contained in the STP-nodes separately. In such a case, in fact, we would neglect the repetition/periodicity information. Temporal consistency checking, thus, proceeds in a top-down fashion, starting from the root of the STP-tree towards its leaves. Basically, the root contains a "standard" STP, so that the Floyd-Warshall's algorithm can be applied to check its consistency.

Thereafter, for each node X in the STP-tree (except the root), we proceed as shown in the algorithm STP_tree_consistency (see Algorithm 1).

Algorithm 1 Algorithm for checking the consistency of a guideline (represented as an STP-tree).

function $STP\_tree\_consistency(X : STPNode$,
$RSpec = (R_1 = \langle nRepetitions_1, I\text{-}Time_1, repConstraints_1, conditions_1\rangle, \ldots,$
$R_n = \langle nRepetitions_n, I\text{-}Time_n, repConstraints_n, conditions_n\rangle)) : STP$

1: check that the repetition/periodicity constraint is well-formed (i.e., that repetitions nest properly)
2: compute Max, i.e. the maximum duration of a single repetition of X according to $RSpec$
3: impose in X that the maximum distance between each pair of points is less or equals Max
4: $X \leftarrow FloydWarshall(X)$
5: if X = INCONSISTENT return INCONSISTENT else return X

STP_tree_consistency takes in input the STP-node that must be checked (i.e. X) and the repetition/periodicity constraint (i.e., the repetition specification in the arc of the STP-tree entering node X), and gives as an output an inconsistency or, in the case of consistency, the local minimal network of the constraints in X considering also the repetition/periodicity constraints. In step 1 it checks whether the repetition/periodicity constraint is well-formed, i.e. if it is consistent when it is taken in isolation (e.g., $I\text{-}Time_2$ must be contained into $I\text{-}Time_1$). In step 2 it computes the maximum duration of a single repetition. This is obtained by considering the time that allows to perform a repetition assuming that all the other repetitions have the minimum possible duration. In step 3 it adds to the STP X the constraints stating that the maximum duration of X must be the computed maximum duration of a single repetition of X. Finally, in step 5 it checks the consistency of the augmented STP X via the Floyd-Warshall's algorithm.

Property. STP_tree_consistency is correct and complete (see [5]). Considering that the number of nesting levels, in the worst case, is less than the number of classes, the algorithm is dominated by step 4, that is $O(C^3)$, where C is the number of actions in the guideline.

Reasoning with the executions of the guideline We have also devised an algorithm for checking the consistency of the execution of a guideline instance with respect to its related guideline. In our work, as in most approaches to clinical guidelines, we suppose that one has full observability of instances (i.e., all the instances of actions which have been executed have been observed and inserted into the knowledge base), and that, for each instance, one knows the corresponding class of actions and/or repetition in the guidelines. The procedure integratedConsistency accepts three parameters: T (the STP-tree that

describes the constraints about classes of actions in the guideline), E (the STP that describes the temporal constraints between the instances of actions – i.e., the actions that have been executed on specific patients), and NOW, that corresponds to the time of the present. The basic idea is to:

(a) check that in the executionSTP there are all and only the instances that the STP-tree predicts to be. Possible missing instances are hypothesized because they may happen in the future;
(b) inherit the repetition/periodicity constraints and the temporal (non-periodic) constraints from the classes to the instances;
(c) propagate the temporal constraints on the executionSTP, thus obtaining the minimal network [4];
(d) check whether the hypothesized instances expected in the future may actually start in the future (i.e., after NOW).

Property. Let us denote with C the number of classes in the STP-tree, with I the number of instances. We have that the complexity of the procedure is $O(max\{I^3, C^3\})$. Also, integratedConsistency is correct and complete as regards consistency checking of the constraints among the instances and among the classes in the STP-tree [5].

3 Clinical Guideline Verification

The verification capabilities concerning the general properties of CIGs and their execution available in the conventional CIG management systems in the literature are usually rather limited. In many cases, such systems do associate only very specific and ad-hoc inferential mechanisms to the knowledge represented in the guideline. To overcome such limitations, the adoption of theorem proving techniques has been proposed within the Protocure European project starting in 2003 [2, 9]. As an alternative of the theorem-proving methodology, the adoption of model-checking techniques has been independently proposed a few years later in the Protocure project [10] and in our project GLARE [11–13], mainly motivated by the simplicity and efficiency of model-checking techniques with respect to the theorem proving approach [14].

Specifically, in our approach we propose a modular solution in which a CIG management system is loosely coupled with a model checker via a translator, which maps any guideline expressed in the formalism of the CIG management system into the formalism of the model-checker. In such a way, the advantages of adopting a CPG management system from one side, and a general-purpose model-checker on the other side are retained and combined. In particular, once the mapping has been defined, any class of properties that can be formalized in the logic of the model checker can be easily verified, without requiring the definition of a new verification software module from scratch. This obviously facilitates a real interaction between the physician examining the CIG and the system itself. Thanks to its modularity, such an approach can be easily implemented, since it does not require any modification to either the CIG management system or the model-checker.

Although our proposal is mostly application-independent, as a proof of concept, we have integrated within the system GLARE [15] a verification tool which models a CIG in Promela, the specification language of the model checker SPIN [16], and verifies the CIG properties to be checked by formalizing them as Linear time Temporal Logic (LTL) formulas.

3.1 Integrating GLARE with SPIN

We have applied the general methodology introduced above in order to couple GLARE with the model checker SPIN. We have implemented a translator which takes in input a GLARE CIG, expressed in the XML format, and transforms it into the corresponding CIG in the Promela language. Analogously, the patient data (in XML) are also translated into the Promela language.

Promela allows a high level model of a distributed system to be defined by modeling each process in an extended pseudo C code, including synchronization primitives and message exchange primitives. Promela provides the usual *if-then-else* and iteration constructs of imperative languages, but it also allows for goto statement (allowing jumps to labels), for the non-deterministic choice construct, as well as for the parallel execution of processes. Processes may share global variables and they also may exchange messages through asynchronous communication channels.

In the following, we briefly describe the general principles we adopt to convert a GLARE CIG into the corresponding agent-based program in the Promela language. First, we describe how a CIG is mapped to a set of interacting processes (called agents henceforth), i.e. to a set of Promela processes and to a set of proper synchronization primitives and message exchange primitives. Then, we shortly describe our translator module.

Guidelines as agents Obviously, the basic object we need to represent in Promela for the purpose of verification is the CIG itself. A CIG can be seen as a set of actions, to be executed in the order specified by a set of control flow primitives. We have mapped each construct (action or control flow primitive) in the CIG to a Promela statement or to a Promela piece of code.

However, CIG execution is a complex phenomenon that cannot be modeled just by representing the CIG per se.

In the following, we propose a possible, more realistic way of capturing the dynamics of the CIG and of its execution environment, based on the idea of modeling a set of processes, whose interaction models the CIG execution itself.

One of the required processes, which we will call agents, is of course the CIG itself. The other agents represent the (human or not) components interacting with the CIG at execution time.

In particular, the Database agent has to be represented. Actually, patient's characteristics need to be specified, and, rather naturally, we characterize a patient by relying on her data, which are typically maintained in the clinical

database. The Database agent thus provides data on demand, and is able to store new data values.

Updated data values are sometimes obtained from additional sources (e.g. from the hospital laboratory service). We have generically modeled such sources and services by means of a further agent, called Outside world.

Last but not least, CIG execution is performed by a physician; therefore, the physician's behavior needs to be modeled as an agent as well. In particular, we have identified two main tasks that the Physician agent is expected to cover when applying a CIG to a specific patient. Obviously, it is required to make decisions, i.e. it has to select exactly one diagnosis or therapy, among a set of alternative ones. Moreover, it has to evaluate data recency and reliability: if a data value, extracted from the database, is judged as unreliable or not up-to-date (i.e. too old), the Physician agent has to rise the problem, thus triggering the generation of a newer data value from the outside world.

In summary, the model of the distributed system we propose to simulate CIG execution can be described by the interaction among the following agents, interpreted as Promela processes:

1. the Guideline agent, which models the overall behavior of the CIG;
2. the Database agent, which models the behavior of the patient database, allowing for data insertion and retrieval;
3. the Outside agent, which represents the outside world and provides up to date values for patient data (together with the time of their measurement) when they are not already available in the database or are evaluated as being not reliable by the physician. It also stores data in the database, and simulates the execution of actions by reporting their success or failure;
4. the Physician agent, which interacts with the CIG by evaluating the patient data, choosing among the different alternative feasible paths as a physician would do, and judging data reliability. Observe that we model the Physician agent as a non-deterministic process, since it is not possible to know a-priori all the possible choices of physicians in all the possible situations. We therefore model the uncertainty about the choice of physicians using non-determinism: from the point of view of the simulation, choices are taken randomly by the Physician agent.

The translator As explained above, we have defined a translator which takes a set of XML documents representing any GLARE CIG and automatically transforms them into the corresponding CIG in the language Promela.

A CIG in GLARE is a hierarchical graph, in which it is possible to have composite actions (i.e. plans), which can be defined in terms of their components via the has-part relation. In the XML document such a structure is maintained. Thus, the translator works as a top-down parser.

In particular, the translator takes in input a graph defined as a couple $\langle N, E \rangle$ (where N is the set of nodes and E is the set of edges), which is the XML document representing the CIG, and a vocabulary V, which contains the medical data information. To make the translation, the parser visits the graph twice. The

first time it makes a preprocessing in order to obtain the data concerning the requests of information.

In the second step, the parser visits the graph for the second time, in order to build the agents which model the CIG behavior.

3.2 CIG verification in SPIN

After the translation, SPIN can be used in order to "reason" about the guidelines. In particular, verification can be managed by expressing properties in LTL, and giving them in input to SPIN, together with the representation of a guideline obtained through the translation process. SPIN translates each Promela process into a finite automaton, and the global behaviour of the system is obtained by computing an asynchronous interleaving product of automata. The resulting automaton represents the global state space of the system (the model containing all the possible executions - runs - of the CIG) and can be built on-the-fly during the verification process. The correctness claims, that have to be checked on the model of the system, are then specified as temporal logic formulas in LTL. Given a property (specification) as an LTL formula, SPIN verifies if the property is true on all the executions of the system. Namely, each run of the system is regarded as a linear temporal model, on which the truth of the property is verified from the initial state.

As a matter of fact, temporal logics such as LTL allow one to express a wide range of formulas. Such an expressiveness and generality motivates a deeper analysis of what kinds of properties, expressible in LTL, are useful in the CIG context. We show some examples, dividing the properties on the basis of the CIG life-cycle phases. Specifically, we single out three main phases (namely, (1) design and acquisition, (2) contextualization, and (3) execution), and we highlight how verification can be fruitfully exploited in each phase.

For the sake of exposition, we describe the properties to be verified by distinguishing two components: (1) a quantifier on "runs": $\forall$, stating that we verify if the property holds on all the runs, and $\exists$, stating that we look for one run satisfying the property; (2) an LTL formula. In the following we assume that the variable "done" in each state is set to the action performed in that state.

Design and acquisition CIGs are usually defined by a national or international committee of specialists, and can be acquired into a computer-based system, usually through a cooperation between some specialists and some knowledge engineers. In such a phase, verification through model checking is useful in order to take into account at least two different classes of properties, namely structural properties and medical validity properties. In particular:

(i) **Structural properties** concern the existence of the appropriate clinical requirements. These properties regard the actions, conditions and paths of actions in the CIG considered "per se", without any reference to the specific context of execution and to the specific patients on which the CIG will be applied, and are relevant in order to ensure the appropriate management of any patients.

Example: verify that any run contains antibiotic treatment (community acquired pneumonia guideline)

$$< \forall run, \diamond(done = antibiotic\_treatment) >$$

Comment: The property evaluates to true if all possible runs contain a state in which an antibiotic treatment is administered.
Relevance: The antibiotic treatment is mandatory in the case of community acquired pneumonia.

(ii) **Medical validity properties** concern both the exclusion of dangerous treatments and the inclusion of the most appropriate treatments for the considered class of patients. These properties are relevant in order to ensure best practice.

Example: verify that whenever hepatic encephalopathy is present, diuretics are not administered (ascites guideline)

$$< \forall run, liver\_state = encephalopathy \rightarrow \Box(done \neq diuretics\_administration) >$$

Comment: Diuretics are contraindicated in hepatic encephalopathy.
Relevance: Diuretics can worsen the liver perfusion and precipitate the encephalopathy or worsen its severity.

Both structural and medical validity properties are verified during the acquisition phase, in which both medical experts and knowledge engineers are usually involved. Specifically, medical experts can identify the structural and validity properties that are relevant for the CIG under consideration, and knowledge engineers can formulate and run the corresponding verifications, reporting the results to the experts. In case the checks show that a desired property does not hold, the domain experts should identify the appropriate corrections to the CIG, which will be modified accordingly, in cooperation with the knowledge engineers.

Contextualization Once a CIG has been defined and acquired (e.g., by a national or international committee), it has to be applied to several different local structures (e.g., hospitals). Unfortunately, in several cases, the original CIG is too "general" to be applied on any specific environment. For instance, depending on the local availability of resources, certain actions of a general CIG cannot be executed in specific contexts (e.g., small hospitals). A phase of contextualization is thus usually needed: when a new CIG is introduced in a hospital, the medical personnel can use verification (possibly in cooperation with knowledge engineers) in order to identify which resources the CIG (or specific paths of the CIG itself) requires. Specifically:

(iii) **Contextualization properties** concern the resources needed for the CIG execution and can be checked to adapt the CIG to locally available resources.

Example: verify that there is a run in which the CT scanner is not used (ischemic stroke guideline)

$$< \exists run, \Box done \neq TC >$$

Comment: If this condition holds, the GL (or, at least a part of it) can be applied also in hospitals where the case the CT scanner is not available.
Relevance: The CT scanner is very important in some cases but not always accessible.

The results of such verifications can be used for modifying the original CIG, or for improving the hospital resources, in order to conform the hospital to the CIG requirements (to grant the best practice). In the last case, the intervention of administrator personnel is also necessary.

Execution Finally, the acquired and contextualised CIGs are used in clinical practice. In such a case, a specific user-physician selects and applies a specific CIG to a specific patient. Verification is a crucial support also in such a phase:

(iv) **Properties concerning the application of a CIG to a specific patient** allow to check which are the best actions (as indicated in the CIG) to be executed on the patient at hand, on the basis of the patients status and symptoms; they also allow to check whether the CIG (or some specific path of it) contains the specific actions which the user-physician expects to be necessary for the patient at hand.

Example: verify that there is a treatment in which growth factors are administered, when leukopenia appears (lymphoma treatment guideline)

$$< \exists run, \Box(leukopenia\_value = present \rightarrow \Diamond\,(done = growth\_factors\_administration)) >$$

Comment: The growth factors administration can positively reduce the duration of the leukopenia and the risk of infections.
Relevance: If leukopenia is not severe, there are also alternative treatments to the administration of growth factors (e.g., expectant treatment and monitoring). That is why we check the existence of one run in which (in a given status) growth factors are administered, without forcing that they are administered in all runs (in contrast with the verification above).

4 Probabilistic Verification

While many of the logical verification methods that have been proposed can be used to verify existing guideliens, a possible shortcoming of the logical methods is that they cannot deal with uncertainty stemming from the use of scientific evidence. On the other hand, many probabilistic methods do not have the representational benefits of logic for modelling temporal constraints between tasks.

In the last few years, there has been a surge of interest in the field of statistical relational learning [17]. In this endeavour, many probabilistic logics have been developed. We believe that these kind advances provide the right ingredients to represent and reason with such heterogeneous medical knowledge.

We think this type of probabilistic verification could particularly be important during the development of a CIG. In this section, we will first introduce a language that can be used to represent guidelines and a probabilistic knowledge base. After this, we show that such a language may be used to represent

a guideline. Finally, we illustrate the approach by means of an example in the development of a hypothetical guideline for diabetes mellitus type 2.

4.1 Causal probabilistic decision logic

We use CP-logic as a starting point, which we will briefly introduce. CP-logic theories consist of a multi-set of causal probabilistic laws (CP-laws), which are statements of the form:

$$\forall x\colon (h_1 : \alpha_1) \vee \cdots \vee (h_n : \alpha_n) \leftarrow b_1, \ldots, b_m \tag{1}$$

where the $\alpha_1 : [0, 1]$ are probabilities with $\sum \alpha_i \leq 1$, $n \geq 1$, and $m \geq 0$. In this formula, h_i and b_j are atoms, that is, expressions of the form $p(t_1, ..., t_m)$ in which p/m is the name of a predicate of arity m and t_i are terms, i.e., constants or variables. We call the set of all $(h_i : \alpha_i)$ the head of the law, and the conjunction of literals b_i the body of the law. We also refer to all h_i as consequences, and to b_i as conditions. If the head contains only one atom $h : 1$, we may write it as h. Informally, the law states that in case the body is true, then at most one of the consequents becomes true, i.e., a consequent is caused by the body. The probabilities in the consequents reflect the probability that the body causes the consequent to become true.

The semantics of CP-logic relies on the notion of a Herbrand interpretation. This is essentially a set of ground atoms that can be constructed using the constant and predicate symbols occurring in the theory. We shall denote Herbrand interpretations by M and we shall write $M \models \varphi$ if the logical formula φ satisfies the interpretation M. Moreover, for simplicity of presentation, we assume a finite set of constants and also that all laws are grounded, i.e., each law is replaced by the set of laws where the variables are replaced by constants. For more details on notions of first-order logic and logic programming, we refer to [18].

CP-logic was designed as a probabilistic logic for modelling causal processes. Actions, too, can be incorporated into these processes, in which case CP-logic requires that the actions that agents take when the body holds is modelled using probabilities. In some cases, one wants to abstract from such actions, for example, to abstract from scheduling decisions when reasoning about concurrent systems. In other situations, there is no probabilistic information about the behaviour of agents, e.g., the course of action of physicians. To be able to model this, we introduce non-determinism into the CP-logic models by adding causal decision laws (CD-laws) to CP-logic. The resulting language is called Causal Probabilistic Decision Logic (CPDL).

Causal decision laws (CD-laws) are of the form:

$$\forall x\colon h_1 \vee \cdots \vee h_n \leftarrow b_1, \ldots, b_m \tag{2}$$

with $n \geq 1$ and $m \geq 0$, which can be seen as CP-laws without any probabilities attached to the elements in the head of the clause. The intuitive reading is also similar to CP-laws, i.e., $b_1, \ldots, b_m$ causes one of the heads, but in this case non-deterministically. That is, again exactly one of the heads is caused by the body, but we do not know which one and also do not know the probabilities.

To obtain a probability distribution for CPDL, the nondeterminism has to be resolved. For this we introduce a policy, which is a function π which maps each ground CD-law to one of its heads. For this function it holds that if $\pi(R) = h_i$, then R is a CD-law of the form:

$$h_1 \vee \cdots \vee h_n \leftarrow b_1, \ldots, b_m.$$

and $1 \leq i \leq n$. The intended semantics is that if $(b_1, \ldots, b_n)$ holds, and $\pi(R) = h_i$, then h_i becomes true. Therefore different groundings will produce different choices (as in CP-logic).

In [19], the semantics of this CP-logic is presented by relating the set of laws to a possible probabilistic causal process. We briefly introduce the semantics of CPDL in the spirit of CP-logic. Consider a CPDL theory T, a policy π, a Herbrand interpretation M, and a grounded law R_k where $R_k = \mathit{head} \leftarrow \mathit{body}$. If R_k is a CP-law and $M \models \mathit{body}$, then the law can be applied for $h_i : \alpha_i \in \mathit{head}$, which we denote by $M \longrightarrow_{\alpha_i} M \cup \{h_i\}$. A CD-law can be applied if also $\pi(R_k) = h_i$ holds, which is denoted by $M \longrightarrow M \cup \{h_i\}$. We then write $M \longrightarrow_p^* M'$ if there exists a chain of applications of laws from M to M' such that each ground law has been applied at most once, no other laws can be applied, and $p = \prod_i \alpha_i$ where α_i are all the probabilities of the applied CP-laws. CPDL defines a joint probability distribution over Herbrand interpretations given a interpretation M by $P_\pi(M') = \sum_{M \longrightarrow_p^* M'} p$. Note that a uniform probability distribution over heads in a CP-law is quite different from a CD-law. If a theory consists of CP-laws, then this models a unique probability distribution. For theories that contain CD-laws, each policy defines a possibly different probability distribution over the Herbrand interpretations.

As a first-order language, CPDL is of course sufficiently rich to model discrete-time models. For representation, we propose a small syntactical extension to the language, which we call CPDTL (Causal Probabilistic Decision Time Logic). We introduce a predicate $\overrightarrow{\cdot}$ which denotes a transition, i.e., $\overrightarrow{a(v)}$ denotes that $a(v)$ holds after a transition. A CPDTL theory contains a set of CPDL laws which may contain $\overrightarrow{\cdot}$, and also a set of *initial laws* of the following form:

$$(a_i(v_{i1})) : \alpha_{i1} \vee \cdots \vee (a_i(v_{in})) : \alpha_{in} \leftarrow \mathbf{M}$$

where $\mathbf{M}$ is called the *starting state*. A CPDTL theory can be mapped to CPDL by replacing all predicates a/m by $a/m+1$ and indexing the predicates by time. Predicates in the initial law get indexed by time 0, i.e., $a_i(v_i)$ becomes $a_i(v_i, 0)$; the atoms $\overrightarrow{a(v)}$ are replaced by $a(v, t+1)$; all other atoms $a(v)$ are replaced by $a(v, t)$. Consider the following example to illustrate this language.

Example 2. Each year, you can decide whether or not to get a flu shot, which affects the chance of becoming infected with influenza (with probability 0.01 when vaccinated, 0.1 when not vaccinated). Influenza might cause other disorders, such as angina (with probability 0.2) and pneumonia (with probability 0.1). Angina causes pneumonia (with probability 0.1), and vice versa (with probability 0.8). Pneumonia might be lethal (with probability 0.01), although there is

also a chance of dying of other causes (with probability 0.001). This medical knowledge of influenza can be represented in CPDTL as follows.

$$
\begin{aligned}
&\text{state(alive)} \leftarrow \mathbf{M}\\
&\text{vaccine(true)} \vee \text{vaccine(false)} \leftarrow \text{state(alive)}\\
&\text{disorder(influenza)} : 0.1 \leftarrow \text{vaccine(false)}\\
&\text{disorder(influenza)} : 0.01 \leftarrow \text{vaccine(true)}\\
&\text{disorder(angina)} : 0.2 \vee \text{disorder(pneumonia)} : 0.1\\
&\qquad\qquad\qquad\qquad \leftarrow \text{disorder(influenza)}\\
&\text{disorder(pneumonia)} : 0.1 \leftarrow \text{disorder(angina)}\\
&\text{disorder(angina)} : 0.8 \leftarrow \text{disorder(pneumonia)}
\end{aligned}
$$

$$
\begin{aligned}
&\overrightarrow{\text{state(dead)}} : 0.01 \vee \overrightarrow{\text{state(alive)}} : 0.99\\
&\qquad\qquad\qquad\qquad \leftarrow \text{disorder(pneumonia)}\\
&\overrightarrow{\text{state(dead)}} : 0.001 \vee \overrightarrow{\text{state(alive)}} : 0.999\\
&\qquad\qquad\qquad\qquad \leftarrow \text{state(alive)}
\end{aligned}
$$

Note that the choice for vaccination has been represented by a CD-law, whereas the rest of the knowledge is represented by a set of CP-laws.

While it is already obvious in this specification that vaccination increases chances of survival, medical researchers are often more interested in relative measures such as the relative risks and the number needed to treat (NNT), i.e., the number of patients who need to be treated to prevent one additional bad outcome. For example, if vaccination decreases the chance on flu from 0.1 to 0.01, then the NNT is $1/(p_{\max} - p_{\min}) = 1/(0.1 - 0.01) \approx 11$, which means that if 11 people are vaccinated, 10 people are not expected to benefit. The NNT for preventing death, however, is not so clear given the interactions between variables. If, however, we would have the minimal and maximal probability of death, then such measure can be easily computed (similar for other relevant measures). If there is one binary decision variable, then this can be solved by computing the outcome of both decisions, but more generally this approach is not feasible.

4.2 Application to guideline verification

The idea of probabilistic verification is now as follows. CPDTL is expressive enough to formalize a non-deterministic automaton using CD-laws. Hence, in principle, a SPIN model derived from a GLARE CIG as described in section 4.2 can be represented in CPDTL. Furthermore, probabilistic information may be combined with this model. The resulting model can be mapped to a probabilistic automaton that can be used to reason with.

Probabilistic automata Probabilistic automata model discrete-time stochastic systems consisting of a (finite) set of states S, an initial state $s_0 \in S$, a (finite) set of actions A and transition probabilities $\mathbf{P}\colon A \times S \times S \to [0, 1]$ such that for all a and s, $\mathbf{P}(a, s, s')$ is a probability distribution over s'. Furthermore,

we assume a labelling function $L\colon S \to 2^{\mathrm{AP}}$ that labels each state with a set of atomic propositions that are true in that state.

A policy $\hat{\pi}\colon S \to A$ is used to decide which action is taken in a state. Given this, a PA can be interpreted as a joint probability distribution over states and actions indexed by time. It is clear that $P(S_0 = s_0) = 1$. Furthermore, the transition probabilities define the conditional probability $P(S_{t+1} = s' \mid S_t = s, A_t = a) = \mathbf{P}(a, s, s')$. Finally, we can interpret policies as a deterministic probability distribution $P(A_t = a \mid S_t = s) = 1$ if $\hat{\pi}(s) = a$.

Given the assumptions above in addition to a Markov assumption, the joint probability of a path, given a policy $\hat{\pi}$, is the product of all transitions that occur in the path, i.e., $P(\mathbf{M}, s_0, \ldots, s_n) = \prod_{t=1}^{n} \mathbf{P}(\hat{\pi}(s_{t-1}), s_{t-1}, s_t)$. In the formal methods community, standard solvers exist to compute lower and upper bounds on probabilities. For this paper, we use the most well-known probabilistic model checker PRISM [20].

Translation To model a CPDTL theory as a PA, we require that we have a set of attributes *Attr*, with a known domain $dom(a)$ for $a \in Attr$, corresponding to a set of predicates in the theory for modelling a state. It should hold that these attributes in the theory are both mutually exclusive and complete. States can be defined as the set of attributes with particular values. As in many other probabilistic logics, this property is not being checked by the system, but should be ensured by the modeller. However, if this holds, then consider a CPDTL theory T with dynamic attributes *Attr* for which we define a PA $M = \langle S, s_0, A, L, \mathbf{P} \rangle$ such that:

- $s \in S$ iff $s = \mathbf{M}$ or for all $a \in Attr$ there exists a unique $v \in dom(a)$ such that $a(v) \in s$
- $A = \{\pi \mid \pi \text{ a CPDL policy for } T\}$
- $s_0 = \mathbf{M}$
- $L(s) = s$
- For all states $s_i, s_k \in S$ and policies $\pi \in A$:

$$\mathbf{P}(\pi, s_i, s_k) = \begin{cases} P_\pi(\overrightarrow{s_k} \mid s_i) & \text{if } s_i \neq s_k \\ 1 - \sum_{j \neq i} P_\pi(\overrightarrow{s_j} \mid s_i) & \text{if } s_i = s_k \end{cases}$$

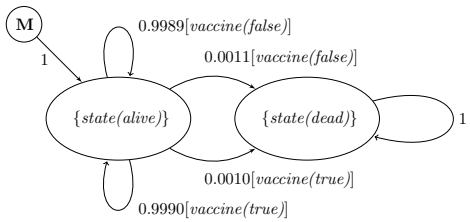
Note that in this definition, the probability of not transitioning to a new state means that you will end up in the same state, which can be seen as a frame axiom. This is a small semantic difference with the original semantics. However, when the transitions are fully specified, i.e., for all π and s_i holds $\sum_k P_\pi(\overrightarrow{s_k} \mid s_i) = 1$, such as in Example 2, then the models will be equivalent in the following sense.

Proposition 1 (fundamental connection PA and CPDTL). *Given a CPDTL theory with state variable s made from the dynamic attributes such that the transitions are fully specified. Let P_{CPDTL} be the corresponding probability distribution and P_{PA} the probability distribution of the corresponding PA. Then*

$$P_{PA}(\boldsymbol{M}, S_0 = s_0, \ldots, S_t = s) = P_{CPDTL}(s(t))$$

Note that for representation, it can be quite useful to have the frame axiom.

Example 3. The knowledge base presented in Example 2 can be graphically represented as a probabilistic automaton as follows:



The state of the PA is defined by the predicate '*state*' and knowledge that models transitions have been abstracted into a single number for each possible policy choice.

4.3 Case study

Using the machinery that has been introduced so far, we investigate a problem of deciding an appropriate treatment for diabetes mellitus type 2. This model is significantly more complex than the influenza example as diabetes is a complicated disease: various metabolic control mechanisms are deranged and many different organ systems may be affected by the disorder. For the individual patient, there is a lot of uncertainty to which extent physiological phenomena occur, which has an impact on the effectiveness of a treatment. We will focus here on a well-known drug called metformin, which is commonly prescribed as the primary oral anti-diabetic, of which the efficacy is known [21]. Moreover, we consider a genetic variation in the encoding of a protein called organic cation transporter 1 (OCT1) which affects the response to metformin [22]. Such knowledge was used to build a set of logical sentences that can be used to explore a simple model of a guideline that recommends treating diabetes with metformin. We answer a number of relevant questions that may come up during the design of such a diabetes guideline. In total, the knowledge base consists of about 50 rules and 10 facts, of which most rules are CD-laws (describing the guideline) and about 10 rules describing background knowledge based on medical literature. For example, it is stated as two CP-laws that there is a 20% chance of carrying the genetic OCT1 variation:

$$\begin{array}{l} \mathit{oct\_variant(true)} : 0.2 \vee \mathit{oct\_variant(false)} : 0.8 \leftarrow \mathbf{M} \\ \overrightarrow{\mathit{oct\_variant(X)}} \leftarrow \mathit{oct\_variant(X)} \end{array}$$

From a practical modelling point of view, variables are not restricted to a finite domain using this logic. For example, in this application, it is convenient to count the number of days that metformin has been given to a patient. This is easily modelled using a counting variables *applied* defined by:

$$\overrightarrow{\mathit{applied(T+1)}} \leftarrow \mathit{state(activated)}, \mathit{applied(T)}$$

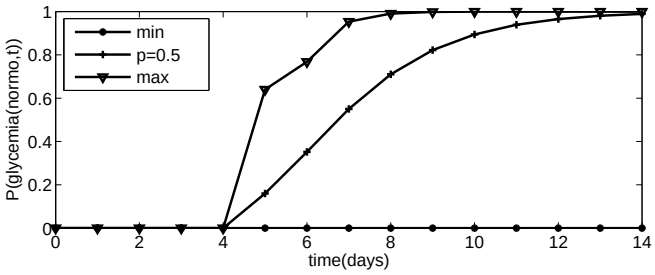


Fig. 2. Probabilistic simulation of metformin application with minimal (min) and maximal (max) probabilities. Furthermore, we consider a model where the physician acts with a probability $p = 0.5$.

which says that, whenever metformin is given to the patient (*state(activated)*), then the counter *applied* is incremented. While this results in an infinite number of states, queries are always related to a time-point. Hence, we can dynamically restrict the domain of T to an appropriate upperbound.

Given such a model, we show a number of queries that explores relevant questions for the practical treatment of diabetes mellitus type 2 patients with metformin.

Question 1: How long should metformin be applied before it can be decided to stop the treatment? There is a trade-off for deciding to stop a treatment: if the treatment with oral anti-diabetics is stopped too early then patients may be injecting themselves with insulin for no good reason; if the treatment is stopped too late, then patients who need treatment with insulin are not treated appropriately. In Figure 2, we plot minimal and maximal probabilities over time. In this case, minimal probabilities are zero, because the guideline did not force the physician to start treatment within a certain time bound, which can be considered a shortcoming of the guideline. We therefore consider another choice where physicians start treatment every day with a certain probability, which in this case is set to a probability 0.5. Furthermore, in Figure 3, we plot a number of dose-response curves for different patients (also with a physician that acts with probability 0.5). For people with an initial low fasting plasma glucose (FPG), the effect of treatment is relatively quick, whereas people with an initial high fasting plasma glucose, the effect is much slower and might not be effective at all even after prolonged treatment. Hence, a recommendation of metformin should take the differences with respect to the baseline fasting plasma glucose into account.

Question 2: What improvement could we gain using genetic information? As the OCT1 protein affects the efficacy of metformin, it might be useful to test whether a patient has a variation in this gene before treatment. In Figure 4, patients are plotted with the same FPG at baseline, but given different evidence with respect to having a variation in the OCT1 protein. On average, patients in this population have a good chance that metformin is effective. However, for

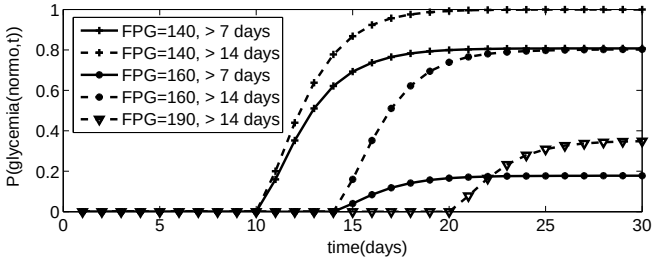


Fig. 3. Probabilistic simulation of metformin application to patients with different fasting blood glucose (FPG) at baseline. Time of metformin application is varied as well.

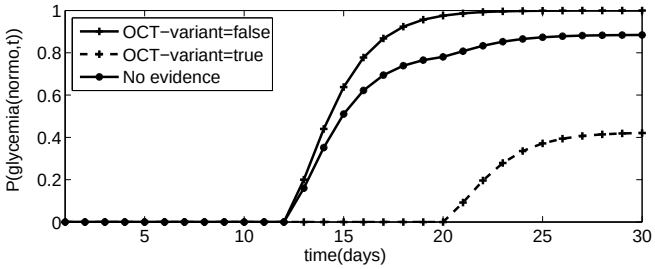


Fig. 4. Probabilistic simulation of metformin application to patients with or without a variation in the OCT1 protein.

the patients with the OCT1-variant, the chance that metformin is effective is rather small and it might be better to prescribe an alternative drug, whereas for patients with a normal OCT1 protein, metformin seems like a good choice. This illustrates that such pharmacogenetics could thus be used for the personalisation of treatments if tests for variations in the OCT1 protein become available.

5 Related works

5.1 Reasoning about Temporal Constraints

Many AI approaches focused their attention to the definition of suitable formalisms to represent time-related phenomena and to reason with them. Besides "logical" approaches (e.g., temporal or non-monotonic logics), starting from the early 80's, many constraint-based approaches have been developed in AI [22]. Such approaches are mostly concerned to define domain-independent knowledge servers which temporal reasoning, in the form of propagation of temporal constraints, can be delegated to, and which can be coupled with other modules (e.g., a planner, or a system which manages guidelines) to solve complex problems.

The aim towards specialization led these approaches to focus on specific classes of constraints (e.g., qualitative constraints such as "A before B", quantitative constraints such as dates, delays and durations) [22], or to devote great

attention to granularities and/or periodic/repeated constraints [23; 24; 25]) or to the integration of different sorts of constraints (e.g., qualitative and quantitative constraints [26]).

In the area of clinical guidelines several interesting approaches have been devised to represent temporal constraints. For instance, GLIF[23] deals both with temporal constraints on patient data elements and with duration constraints on actions and decisions. In PRO*forma* [24], guidelines are modelled as plans, and each plan may define constraints on the accomplishment of tasks, as well as task duration and delays between tasks. Moreover, temporal constructs can also be used in order to specify the preconditions of actions. DILEMMA and PRESTIGE [3] model temporal constraints within conditions. EON [25] uses temporal expressions to allow the scheduling of guideline steps, and deals with duration constraints about activities. Moreover, by incorporating the RESUME system, it provides a powerful approach to cope with temporal abstraction. In EON, the Arden Syntax allows the representation of delays between the triggering event and the activation of a Medical Logic Module (MDL), and between MDLs [26].

A rich ontology to deal with temporal information in clinical trial protocols has been proposed in [27], considering also relative and indeterminate temporal information and cyclical event patterns.

Despite the large amount of work devoted to the representation of temporal constraints, and the very rich and expressive formalisms being identified, little attention has been paid to temporal reasoning. Notable exceptions are represented by the approaches by Shahar [28] and by Duftschmid et al. [29].

In Shahar's approach, the goal of temporal reasoning is not to deal with temporal constraints (e.g., to check their consistency), but to find out proper temporal abstractions to data and properties. Therefore, temporal reasoning is not based on constraint propagation techniques, in fact, e.g., interpolation-based techniques and knowledge-based reasoning are used.

Miksch et al. have proposed a comprehensive approach based on the notion of temporal constraint propagation [28, 29]. In particular, in Miksch et al.'s approach, different types of temporal constraints deriving from the scheduling constraints in the guideline, from the hierarchical decomposition of actions into their components and from the control-flow of actions in the guideline are mapped onto an STP framework [4]. Temporal constraint propagation is used in order to (1) detect inconsistencies, and to (2) provide the minimal constraints between actions. In [29], there is also the claim that (3) such a method can be used by the guideline interpreter in order to assemble feasible time intervals for the execution of each guideline activity. Moreover, advanced visualization techniques are used in order to show users the results of temporal reasoning [30].

5.2 Verification

Our work about model-checking verification has started in the context of the Italian (two-years) project MIUR-PRIN 2003 "Logic-based development and verification of multi-agent systems" whose main objective was the development of logical and computational formalisms for the specification and verification

of agents and their interactions. Our approach to LTL verification of clinical guidelines in SPIN has been described in detail in [31]. [[OK REF A AIMJ????]]

Automatic verification of clinical guidelines has first been explored in [9], where a theorem proving approach is proposed to deal with the problem of protocol verification. This activity has been developed within the European projects Protocure and Protocure II. Here, a medical protocol is modelled in the Asbru language as a hierarchical plan and then it is mapped to a specification in KIV, an interactive theorem prover for higher order logic. Properties are expressed in a variant of Interval Temporal Logic. [2] has provided an evaluation of the feasibility of this approach based on the formalization and verification of the "jaundice" protocol and the "diabetes mellitus" protocol.

In the Protocure II project, model checking techniques for the verification of clinical guidelines have also been explored [10]. In contrast to interactive verification, model checking is fully automatic. In particular, Protocure II exploits CTL model checking and the tool SMV [32]. The Asbru model is translated into the input language of SMV model checker by making use of a suitable abstraction which eliminates time. The compiler takes the algebraic specification of Asbru models in KIV as input and generates an SMV document. CTL model checking is used in the verification of a wide range of properties of guidelines modelled in Asbru, namely structural and medical properties. In particular, in [10] properties of the jaundice protocol are formalized as ACTL formulas (that is, CTL formulas only allowing universal path quantifiers) [33].

The main difference between our approach and Protocure's one is that our approach is based on LTL temporal logic while Protocure's one is based on CTL temporal logic. The adoption of CTL (and ACTL) or LTL model checking allows for the verification of different temporal properties, as CTL and LTL are expressively incomparable (as well as ACTL and LTL).A further difference between our approach and Protocure one is due to the availability in SPIN of a higher-level input language, as compared with the input language of SMV. The fact that Promela is well suited for modelling guidelines as processes interacting with their environment by exchanging messages over channels, substantially simplifies the task of providing a translation of guidelines into Promela code (which does not require intermediate levels of representation), as well as that of interpreting the results of Concerning the type of the properties to be verified, as observed in [10] the model checking approach is well suited for the verification of structural and simple medical properties of the guideline, that normally do not require an incremental verification strategy.

5.3 Probabilistic Verification

Since the last two decades, probabilistic graphical models, PGMs for short, have become the state of the art for knowledge representation involving uncertainty. PGMs, and in particular Bayesian and Markov networks, have been successfully applied to various problem areas, including medicine. There is a considerable body of work (e.g. [34–38]) indicating that Bayesian networks offer a natural and intuitive formalism for constructing clinically relevant models. Unfortunately,

PGMs are unsuitable for capturing knowledge that goes beyond statistical dependence and independence information, like clinical guidelines. In contrast, it has been shown that CP-logic that the probabilistic verification introduced in this chapter is based upon, can also represent various PGMs [19, 39].

With the recent introduction of probabilistic logics more powerful, relational languages for the representation of uncertain knowledge have become available, which are more suitable for dealing with combinations of logical and probabilistic knowledge. For example, there now exist logical versions of Markov networks, called *Markov logic networks* [40], and of Bayesian networks, called *Bayesian logic programs* [41]. Influential is also Poole's independent choice logic [42, 43], in addition to ProbLog [44] and CP-Logic [19]. These probabilistic logics offer a very natural and flexible choice for modelling complex domains involving uncertainty. On the other, all these languages are general probabilistic logics and do not deal with the particular requirements for representing CIGs, in particular temporal information. A notable exception is CPT-L [45], where CP-logic is used for modelling Markov models. The difference to our approach in the fact that in CPT-L each rule determines a transition. In the logic proposed in this chapter, each derivation determines a transition, which allows for richer modelling of the transitions.

The CPDTL logic proposed here can also been as a hierarchical models, with on the top-level a Markov model and in each state transition, a CP-logic program. In this sense, logical hierarchical HMM (LoHiHMM) [46] can be seen as a related appraoach. The difference with this approach is two-fold. In the probabilistic sense, the LoHiHMM is more expressive, as it is a hidden Markov model rather than a Markov chain. On the other hand, the logical representation is weaker. LoHiHMMs are called logical, because each state is abstracted using predicate logic, rather than a propositional state. However, logic is not used to model the transitions, which makes it unsuitable to model the dynamic behaviour using a symbolic language.

6 Conclusions

CIG are assuming an important role in the standardization and optimization of healthcare. In particular, CIG system can be used by physicians as recommendation tools, to provide high-quality medical treatments to patients, on the basis of evidence-based medicine.

However, given the dimensions of CIGs (which may consist of hundreds of inter-related actions), and the large amount of knowledge they contain, verification is important to guarantee the quality of the provided recommendations. As described in this chapter, verification is important at different stages in the CIG life-cycle: during design and acquisition, to check structural and medical validity properties, during contextualization, to support the adoption of general CIGs in specific application contexts, and during execution, to look for the most appropriate treatments of specific patients, possibly considering probabilistic information on treatments outcomes and patient evolutions.

In this chapter, we investigate such issues, proposing a range of methodologies covering the different aspects of verification. The core idea of this chapter is that, given the heterogeneous character of the knowledge contained in CIGs, different forms of verifications should be supported, demanding for an hybrid approach in which different representation formalisms are used (to properly capture different types of knowledge) and different methodologies are devised (to properly reason with the different formalisms). In particular, we focused on three different issues, and methodologies.

Considering the temporal constraints in CIGs, we propose to adopt the classical AI approach to devise a specialized constraint-propagation-based temporal reasoner [47, 48]. Such approaches focus on temporal constraints only, so that specific representation formalisms can be devised, in such a way that correct and complete temporal reasoning can be performed efficiently (in cubic time, in our approach). While such approaches are advantageous when considering temporal constraints only, they are not general enough to deal with more general forms of knowledge, and of verification. We then considered the verification of medical validity properties of CIGs, with emphasis on the temporal evolution of CIGs actions. In such a context, we have proposed the adoption of model-based verification techniques based on temporal logics (LTL), which proved to be well-suited for many medical verification tasks. Last, but not least, we have also considered the probabilistic component of medical knowledge, which is of paramount importance especially when considering the application of CIGs to specific patients, to identify those treatment which, probabilistically speaking, are the best options for them. In such a context, we propose the Causal Probabilistic Decision Time Logic, and show its adequacy to deal with probabilistic verification in the medical context.

References

1. Overhage, J.M., Tierney, W.M., Zhou, X.H.A., McDonald, C.J.: A randomized trial of corollary orders to prevent errors of omission. Journal of the American Medical Informatics Association **4**(5) (1997) 364–375
2. Ten Teije, A., Marcos, M., Balser, M., van Croonenborg, J., Duelli, C., van Harmelen, F., Lucas, P., Miksch, S., Reif, W., Rosenbrand, K., et al.: Improving medical protocols by formal methods. Artificial intelligence in medicine **36**(3) (2006) 193–209
3. Musen, M., Rohn, J., Fagan, L., Shortliffe, E.: Knowledge engineering for a clinical trial advice system: Uncovering errors in protocol specification. Bulletin du cancer **74**(291) (1987) 296
4. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artificial intelligence **49**(1) (1991) 61–95
5. Anselma, L., Terenziani, P., Montani, S., Bottrighi, A.: Towards a comprehensive treatment of repetitions, periodicity and temporal constraints in clinical guidelines. Artificial Intelligence in Medicine **38**(2) (2006) 171–195
6. Egidi, L., Terenziani, P.: A lattice of classes of user-defined symbolic periodicities. In: Temporal Representation and Reasoning, 2004. TIME 2004. Proceedings. 11th International Symposium on. (2004) 13–20

7. Egidi, L., Terenziani, P.: A mathematical framework for the semantics of symbolic languages representing periodic time. In: Temporal Representation and Reasoning, 2004. TIME 2004. Proceedings. 11th International Symposium on. (2004) 21–27
8. Bettini, C., De Sibi, R.: Symbolic representation of user-defined time granularities. In: Temporal Representation and Reasoning, 1999. TIME-99. Proceedings. Sixth International Workshop on. (1999) 17–28
9. Marcos, M., Balser, M., ten Teije, A., van Harmelen, F., Duelli, C.: Experiences in the formalisation and verification of medical protocols. In: Artificial Intelligence in Medicine. Springer (2003) 132–141
10. Bäumler, S., Balser, M., Dunets, A., Reif, W., Schmitt, J.: Verification of medical guidelines by model checking–a case study. In: Model Checking Software. Springer (2006) 219–233
11. Giordano, L., Martelli, A., Terenziani, P., Bottrighi, A., Montani, S.: A temporal approach to the specification and verification of interaction protocols. In: WOA. (2005) 171–176
12. Terenziani, P., Giordano, L., Bottrighi, A., Montani, S., Donzella, L.: Spin model checking for the verification of clinical guidelines. In: ECAI 2006 Workshop on AI techniques in healthcare: evidence-based guidelines and protocols. (2006)
13. Giordano, L., Terenziani, P., Bottrighi, A., Montani, S., Donzella, L.: Model checking for clinical guidelines: an agent-based approach. In: AMIA Annual Symposium Proceedings. Volume 2006., American Medical Informatics Association (2006) 289
14. Halpern, J.Y., Vardi, M.Y.: Model checking vs. theorem proving: a manifesto. Artificial Intelligence and Mathematical Theory of Computation. Academic Press, Inc **212** (1991) 151–176
15. Terenziani, P., Molino, G., Torchio, M.: A modular approach for representing and executing clinical guidelines. Artificial intelligence in medicine **23**(3) (2001) 249–276
16. Holzmann, G.J.: The model checker spin. Software Engineering, IEEE Transactions on **23**(5) (1997) 279–295
17. Getoor, L., Taskar, B., eds.: Introduction to Statistical Relational Learning. Adaptive Computation and Machine Learning. MIT Press (2007)
18. Lloyd, J.: Foundations of Logic Programming, 2nd Edition. Springer (1987)
19. Vennekens, J., Denecker, M., Bruynooghe, M.: CP-logic: A language of causal probabilistic events and its relation to logic programming. Theory and Practice of Logic Programming **9** (2009) 245–308
20. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic symbolic model checker. In Field, T., Harrison, P., Bradley, J., Harder, U., eds.: Proceedings TOOLS'02. Volume 2324 of LNCS., Springer (2002) 200–204
21. Garber, A., Duncan, T., Goodman, A., Mills, D., Rohlf, J.: Efficacy of metformin in type II diabetes: results of a double-blind, placebo-controlled, dose-response trial. Am J Med **103**(6) (1997) 491–507
22. Shu, Y., Sheardown, S., Brown, C., Owen, R., Zhang, S., Castro, R., Ianculescu, A., Yue, L., Lo, J., Burchard, E., Brett, C., Giacomini, K.: Effect of genetic variation in the organic cation transporter 1 (OCT1) on metformin action. J Clin Invest **117** (2007) 1422–1431
23. Peleg, M., Boxwala, A.A., Ogunyemi, O., Zeng, Q., Tu, S., Lacson, R., Bernstam, E., Ash, N., Mork, P., Ohno-Machado, L., et al.: Glif3: the evolution of a guideline representation format. In: Proceedings of the AMIA Symposium, American Medical Informatics Association (2000) 645
24. Fox, J., Johns, N., Rahmanzadeh, A.: Disseminating medical knowledge: the PRO*forma* approach. Artificial intelligence in medicine **14**(1) (1998) 157–182

25. Musen, M.A., Tu, S.W., Das, A.K., Shahar, Y.: EON: A component-based approach to automation of protocol-directed therapy. Journal of the American Medical Informatics Association **3**(6) (1996) 367–388
26. Sherman, E.H., Hripcsak, G., Starren, J., Jenders, R.A., Clayton, P.: Using intermediate states to improve the ability of the arden syntax to implement care plans and reuse knowledge. In: Proceedings of the Annual Symposium on Computer Application in Medical Care, American Medical Informatics Association (1995) 238
27. Weng, C., Kahn, M., Gennari, J.: Temporal knowledge representation for scheduling tasks in clinical trial protocols. In: Proceedings of the AMIA Symposium, American Medical Informatics Association (2002) 879
28. Shahar, Y.: A framework for knowledge-based temporal abstraction. Artificial intelligence **90**(1) (1997) 79–133
29. Duftschmid, G., Miksch, S., Gall, W.: Verification of temporal scheduling constraints in clinical practice guidelines. Artificial intelligence in medicine **25**(2) (2002) 93–121
30. Kosara, R., Miksch, S.: Visualization methods for data analysis and planning in medical applications. International Journal of Medical Informatics **68**(1) (2002) 141–153
31. Bottrighi, A., Giordano, L., Molino, G., Montani, S., Terenziani, P., Torchio, M.: Adopting model checking techniques for clinical guidelines verification. Artificial Intelligence in Medicine **48**(1) (2010) 1–19
32. McMillan, K.L.: Symbolic model checking. Springer (1993)
33. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. MIT press (1999)
34. Andreassen, S., Riekehr, C., Kristensen, B., Schonheyder, H., Leibovici, L.: Using probabilistic and decision-theoretic methods in treatment and prognosis modeling. Artificial Intelligence in Medicine **15**(2) (1999) 121–134
35. Lucas, P., de Bruijn, N., Schurink, K., Hoepelman, I.: A probabilistic and decision-theoretic approach to the management of infectious disease at the icu. Artificial Intelligence in Medicine **19**(3) (2000) 251–279
36. Paul, M., Andreassen, S., Tacconelli, E., Nielsen, A., Almanasreh, N., Frank, U., Cauda, R., Leibovici, L.: Improving empirical antibiotic treatment using treat, a computerized decision support system: cluster randomized trial. Journal of Antimicrobial Chemotherapy **58** (2006) 1238–1245
37. Zalounina, A., Paul, M., Leibovici, L., Andreassen, S.: A stochastic model of susceptibility to antibiotic therapy–the effects of cross-resistance and treatment history. Artificial Intelligence in Medicine **40**(1) (2007) 57–63
38. Velikova, M., Samulski, M., Lucas, P., Karssemeijer, N.: Improved mammographic cad performance using multi-view information: a Bayesian network framework. Physics in Medicine and Biology **54** (2009) 1131–1147
39. Hommersom, A., Ferreira, N., Lucas, P.: Integrating logical reasoning and probabilistic chain graphs. In: Proceedings of ECML/PKDD 2009. Volume 5781 of LNAI., Springer, Berlin-Heidelberg (2009) 548–563
40. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1-2) (2006) 107–136
41. Kersting, K., De Raedt, L.: Towards combining inductive logic programming with Bayesian networks. In: Proceedings of the Eleventh International Conference on Inductive Logic Programming, Springer (2001) 118–131
42. Poole, D.: The independent choice logic for modelling multiple agents under uncertainty. Artificial Intelligence **94**(1-2) (1997) 7–56

43. Poole, D.: The independent choice logic and beyond. In de Raedt, L., Frasconi, P., Kersting, K., Muggleton, S., eds.: Probabilistic Inductive Logic Programming: Theory and Application. Number 4911 in LNAI, Berlin, Springer (2008)
44. Kimmig, A., Demoen, B., De Raedt, L., Santos Costa, V., Rocha, R.: On the implementation of the probabilistic logic programming language problog. Theory and Practice of Logic Programming (2010)
45. Thon, I., Landwehr, N., De Raedt, L.: A Simple Model for Sequences of Relational State Descriptions. In: Proceedings of the 19th European Conference on Machine Learning. (2008) 506–521
46. Natarajan, S., Bui, H., Tadepalli, P., Kersting, K., Wong, W.K.: Logical hierarchical hidden markov models for modeling user activities. In: ILP '08: Proceedings of the 18th international conference on Inductive Logic Programming, Berlin, Heidelberg, Springer-Verlag (2008) 192–209
47. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11) (1983) 832–843
48. Vila, L.: A survey on temporal reasoning in artificial intelligence. AI Commun. **7**(1) (1994) 4–28